Sorting from Noisy Information

Mark Braverman*
Microsoft Research New England

Elchanan Mossel†
Statistics and Computer Science, U.C. Berkeley and
Mathematics and CS, Weizmann Institute, Rehovot, Israel

October 7, 2009

Abstract

This paper studies problems of inferring order given noisy information. In these problems there is an unknown order (permutation) π on n elements denoted by $1, \ldots, n$. We assume that information is generated in a way correlated with π. The goal is to find a maximum likelihood π^* given the information observed. We will consider two different types of observations: noisy comparisons and noisy orders.

- *Noisy Orders (also called the Mallow's model).* Given the original permutation π, the probability of a permutation σ being generated is proportional to $e^{-\beta d_K(\sigma,\pi)}$. In other words, the probability is inverse exponential in the Kemeny distance of π from σ, which is the number of pairs ordered in π differently from σ:

$$d_K(\pi,\sigma) = \#\{(i,j) : \ \pi(i) < \pi(j) \text{ and } \sigma(i) > \sigma(j)\}.$$

  We assume that we are given $\sigma_1, \ldots, \sigma_r$ that are generated independently conditioned on π.

- *Noisy Comparisons.* The input is the status of $\binom{n}{2}$ queries of the form $q(i,j)$, for $i < j$, where $q(i,j) = +(-)$ with probability $1/2 + \lambda$ if $\pi(i) > \pi(j)(\pi(i) < \pi(j))$ for all pairs $i \neq j$, where $\lambda > 0$ is a constant. It is assumed that the errors are independent. More generally, the input may be any collection of independent biased signals on the order relationship between pairs of elements.

*Part of this work was done while the author was a graduate student at the University of Toronto, supported by and NSERC CGS scholarship. Part of the work was done while the author was visiting IPAM, UCLA.

†Supported by an Alfred Sloan fellowship in Mathematics, by NSF grants DMS-0528488 and DMS-0548249 (CAREER), by DOD ONR grant N0014-07-1-05-06 and by ISF grant 1300/08. Part of this work was done while the author was visiting IPAM, UCLA.

In this paper we present polynomial time algorithms for solving both problems with high probability. For noisy orders the running time of the algorithm is $n^{1+O((\beta r)^{-1})}$, and for noisy comparisons the algorithm runs in time $n^{O(\lambda^{-3-\varepsilon})}$. Both algorithms have $O(n \log n)$ query complexity (with the constant depending on λ, β and r).

As part of our proof we show that for both models the maximum likelihood solution π^* is close to the original permutation π. More formally, with high probability it holds that

$$\sum_i |\pi(i) - \pi^*(i)| = \Theta(n), \quad \max_i |\pi(i) - \pi^*(i)| = \Theta(\log n).$$

Our results are of interest in applications to ranking, such as ranking in sports, or ranking of search items based on comparisons by experts.

1 Introduction

We study the problem of sorting in the presence of noise. While sorting linear orders is a classical well studied problem, the introduction of noise creates very interesting challenges. Noise has to be considered when ranking or sorting is applied in many real life scenarios.

A natural example comes from sports. How do we rank a league of soccer teams based on the outcomes of the games? It is natural to assume that there is a true underlying order of which team is better and that the game outcomes represent noisy versions of the pairwise comparisons between teams. Note that in this problem it is impossible to "re-sample" the order between a pair of teams. As a second example, consider experts ranking various items according to their importance. It is natural to assume that the experts' opinions represent a noisy view of the actual order of significance. The question is then how to aggregate this information?

1.1 Aggregating rankings: Mallow's Model

The classical model for noisy permutations was introduced by Mallow [Mal57]. This model is parameterized by a permutation π^* and a real parameter $\beta > 0$. The probability of observing a permutation π is exponentially small in β times the distance between π and π^*. More formally, given the original permutation π^*, the probability of a permutation π being generated is inverse exponential in the Kemeny distance of π from π^*. The Kemeny distance is the number of pairs ordered in π differently from π^*:

$$d_K(\pi, \pi^*) = \#\{(i,j) : \ \pi^*(i) < \pi^*(j) \text{ and } \pi(i) > \pi(j)\}. \tag{1}$$

Definition 1. *In Mallow's model, the probability of a permutation π is given by*

$$\mathrm{P}[\pi|\pi^*] = \frac{1}{Z(\beta)} e^{-\beta d_K(\pi,\pi^*)}. \tag{2}$$

for a $\beta > 0$ and a normalization constant $Z(\beta)$.

This model has been studied extensively in statistics and has been generalized in a number of ways, see e.g. [Dia88, FV86, FV88].

Our goal is to find the best fit for the permutation π^* given r independent observations $\pi_1, \ldots, \pi_r$ that are distributed according to (2).

Definition 2. *The* Mallow Reconstruction Problem *(MRP) is the problem of finding a π^* maximizing the quantity*

$$\prod_{k=1}^{r} \Pi[\pi_k|\pi^*] = \frac{1}{Z(\beta)^r} e^{-\beta \sum_{k=1}^{r} d_K(\pi_k, \pi^*)},$$

or equivalently minimizing

$$d(\pi^*) := \sum_{k=1}^{r} d_K(\pi_k, \pi^*). \tag{3}$$

The optimization problem without any assumptions on the generating process is NP-hard [BTT89]. On the other hand, a number of heuristics were suggested in the statistical literature for solving the problem [FV90, CSS99, MPPB07]. None of these heuristics have a guarantee to find the correct permutation even assuming the permutations are generated from the model.

In one of our main results we will show that the MRP problem can be solved in polynomial time, that approaches linear time as r increases.

1.2 Aggregating noisy comparisons

We next define a second model for noisy sorting. In this model the noise is applied to each pairwise comparison. In other words, for each pair, the correct order is observed with some probability greater than 1/2.

1.2.1 The sorting model: Noisy Signal Aggregation

We will consider the following probabilistic model of instances. There will be n items denoted $1, \ldots, n$. There will be a *true order* given by a permutation π on $1, \ldots, n$. For two elements $i, j \in [n]$ we write $i <_\pi j$ if $\pi(i) < \pi(j)$.

The algorithm will have access to $\binom{n}{2}$ signals defined as follows.

For each unordered pair $\{a, b\}$, it receives a signal $s_{a,b} = s_{b,a}$. The signal distribution $\mathcal{D}$ depends on whether $a < b$ or $b < a$:

$$\mathcal{D} = \begin{cases} \mathcal{D}_{a<b} & \text{if } \pi(a) < \pi(b), \\ \mathcal{D}_{b<a} & \text{if } \pi(b) < \pi(a). \end{cases} \tag{4}$$

We assume that the signals are independent conditioned on the true order. In other words, for any set $S = \{(i_1, j_1), (i_2, j_2), \ldots, (i_k, j_k)\}$ of unordered pairs, such that $(a, b) \notin S$, and a vector of signals $s = (s_{i_1,j_1}, s_{i_2,j_2}, \ldots, s_{i_k,j_k})$,

$$\mathcal{D}\left[s_{a,b} = \cdot \mid \pi, s\right] = \mathcal{D}\left[s_{a,b} = \cdot \mid 1_{\pi(a)<\pi(b)}\right].$$

The goal of *Noisy Signal Aggregation (NSA)* problem defined below is to find a permutation π that is most consistent with the signals.

Definition 3. *Given the signals $s_{i,j}$ for all pairs $\{i,j\} \in [n]$, the Noisy Signal Aggregation is the maximum likelihood permutation π, assuming uniform prior. In other words, π maximizes the quantity*

$$\mathrm{P}[\{s_{i,j}\} \mid \pi] = \prod_{i,j:i<_\pi j} \mathcal{D}_{i<j}(s_{i,j}). \tag{5}$$

Given a signal $s_{a,b}$, assuming uniform prior we have

$$\frac{\mathrm{P}[a <_\pi b \mid s_{a,b}]}{\mathrm{P}[b <_\pi a \mid s_{a,b}]} = \frac{\mathcal{D}_{a<b}(s_{a,b})}{\mathcal{D}_{b<a}(s_{a,b})}. \tag{6}$$

We associate a score $q(a<b)$ with the decision to rank a below b as the log of this ratio:

$$q(a<b) := \log \frac{\mathcal{D}_{a<b}(s_{a,b})}{\mathcal{D}_{b<a}(s_{a,b})}. \tag{7}$$

Obviously, $q(b<a) = -q(a<b)$. Note that by Gibbs' inequality $\mathrm{E}[q(a<b)|\pi(a)<\pi(b)] \geq 0$. The NSA problem thus can be rephrased as the following problem.

Proposition 4. *The NSA Problem is equivalent to the problem of finding a σ that maximizes the total score*

$$s_q(\sigma) := \sum_{i<_\sigma j} q(i<j). \tag{8}$$

We will discuss several NSA models. The simplest one is defined as follows.

Definition 5. *The* Simple Noisy Sorting Aggregation*(SNSA) problems with parameter λ is a NSA problem where $s_{a,b} \in \{+,-\}$ for all a,b and*

$$\mathcal{D}_{a>b}(+) = \frac{1}{2}+\lambda, \quad \mathcal{D}_{a>b}(-) = \frac{1}{2}-\lambda, \tag{9}$$

$$\mathcal{D}_{a<b}(+) = \frac{1}{2}-\lambda, \quad \mathcal{D}_{a<b}(-) = \frac{1}{2}+\lambda.$$

Our results showing that the SNSA can be solved efficiently are presented in section 1.3. The results are also extended to a much more general family of NSA problems.

1.2.2 Related Sorting Models and Results

It is natural to consider the problem of finding a ranking σ that minimizes the score $s_q(\sigma)$ where the input q takes only the values of ± 1 (a relation between every pair), and there are no probabilistic assumptions on the input. This problem, called the *feedback arc set*

problem for tournaments is known to be NP hard [ACN05, Alo06]. However, it does admit PTAS [KMS07] achieving a $(1+\epsilon)$ approximation for

$$-\frac{1}{2}\left[s_q(\sigma) - \binom{n}{2}\right].$$

in time that is polynomial in n and doubly exponential in $1/\epsilon$. The results of [KMS07] are the latest in a long line of work starting in the 1960's and including [ACN05, Alo06]. See [KMS07] for a detailed history of the feedback arc set problem.

A problem that is in a sense easier than NSA is the problem where repetitions are allowed in querying. In this case it is easy to observe that the original order may be recovered in $O(n \log^2 n)$ queries with high probability. Indeed, one may perform any of the standard $O(n \log n)$ sorting algorithms and repeat each query $O(\log n)$ times in order to obtain the actual order between the queried elements with error probability n^{-2} (say). More sophisticated methods show that in fact the true order may be found in query complexity $O(n \log n)$ with high probability [FPRU90], see also [KK07].

Remark 6. *Some of our results on the SNSA problem appeared as an extended abstract in [BM08].*

1.3 Main Results

1.3.1 Mallow Reconstruction Problem

For the Mallow Reconstruction Problem our main result is that the problem can be solved in time that tends to linear as r increases beyond $1/\beta$. Formally, we prove the following:

Theorem 7. *There exists a randomized algorithm such that if $\pi_1, \ldots, \pi_r$ be rankings on n elements independently generated by Mallow's model with parameter $\beta > 0$, and let $\alpha > 0$. Then a maximum probability order π^m can is computed in time*

$$T(n) = O\left(n^{1+O\left(\frac{\alpha}{\beta r}\right)} \cdot 2^{O\left(\frac{\alpha}{\beta}+\frac{1}{\beta^2}\right)} \cdot \log^2 n\right).$$

and error probability $< n^{-\alpha}$. In particular, the algorithm tends to almost linear as r grows.

1.3.2 Simple Noisy Signal Aggregation

For the Simple Noisy Signal Aggregation problem, our main result is the following.

Theorem 8. *For any $\lambda > 0$ and $\alpha > 0$ there exists a randomized algorithm that except with probability at most $n^{-\alpha}$ finds an optimal solution to the Simple Noisy Signal Aggregation (SNSA) with parameter λ in time $n^{O((\alpha+1)\lambda^{-4})}$.*

1.3.3 General Noisy Aggregation

Our results extend to more general models of NSA aggregations which we now discuss. In order for our aggregative reconstruction to work, we will need two properties from the signal distributions.

Definition 9. *We say that a collection of distributions $\mathcal{D}_{a<b}, \mathcal{D}_{b<a}$ is* strongly γ-biased *if*

(a) For every $\frac{\log n}{\gamma} < m \leq n$, and for any m different $\mathcal{D}_{a_k,b_k}$ such that $a_k <_\pi b_k$ for at least $2/3$ of the k's:

$$\mathrm{P}\left[\sum_{k=1}^{m} q(a_k < b_k) > 0\right] > 1 - 2^{-\gamma m}. \tag{10}$$

(b) There is a constant A such that for any A different $\mathcal{D}_{a_k,b_k}$ such that $a_k < b_k$ holds for all the k's,

$$\mathrm{P}\left[\sum_{k=1}^{A} q(a_k < b_k) > 0\right] > 1 - 10^{-3}. \tag{11}$$

Under these conditions we prove the following.

Theorem 10. *For any $\gamma > 0$ and $\alpha > 0$ there exists a randomized algorithm that except with probability at most $n^{-\alpha}$ finds an optimal solution to the Noisy Signal Aggregation (NSA) problem on strongly γ-biased signals in time $n^{\tilde{O}((\alpha+1)\gamma^{-3})}$.*

In the statement above and throughout the paper $\tilde{O}(\cdot)$ signifies order of magnitude up to logarithmic corrections in the variables in the expression inside the $\tilde{O}(\cdot)$. A key ingredient in the proof of Theorem 10 is the following.

Theorem 11. *Consider the NSA problem on strongly γ-biased signals and let π be the* true *order and σ be any optimal order. Let $\alpha > 0$. Then there exist constants $c_1(\alpha, \gamma)$ and $c_2(\alpha, \gamma)$ such that except with probability $O(n^{-\alpha})$ the following inequalities hold:*

$$\sum_{i=1}^{n} |\sigma(i) - \pi(i)| \leq c_1 n, \tag{12}$$

$$\max_i |\sigma(i) - \pi(i)| \leq c_2 \log n. \tag{13}$$

Extending the techniques of [FPRU90] it is possible to obtain the results of Theorem 10 with low sampling complexity. More formally,

Theorem 12. *There is an implementation of a sorting algorithm with the same guarantees as in Theorem 10 and whose sampling complexity is $C\, n \log n$ where $C = C(\alpha, \gamma, A)$.*

In fact, Theorems 10 and 11 only require condition (a) from Definition 9. Condition (b) is only used to establish low sampling complexity in Theorem 12. We note that without condition (b) Theorem 12 holds with sampling complexity of $O(n \log^2 n)$ rather than $O(n \log n)$.

We briefly note that from Azuma inequality it follows that

Claim 13. *SNSA distributions (9) with parameter λ are strongly γ biased with $\gamma = \Omega(\lambda)$.*

Therefore Theorem 8 follows from Theorems 10 and 12. More generally we have the following claim that gives a large set of strongly γ-biased distributions:

Claim 14. *Consider the NSA problem where there exists a constant C such that for all a, b the functions $q(a, b)$ and $q(b, a)$ are bounded by C and*

$$\mathrm{E}[q(a < b)|\pi(a) < \pi(b)] > \lambda \quad \textit{and} \quad \mathrm{E}[q(b < a)|\pi(b) < \pi(a)] > \lambda.$$

Then the distributions $D_{a<b}, D_{a>b}$ are strongly-γ biased $\gamma = \Omega(\lambda/C)$.

1.4 Techniques

1.4.1 Mallow Reconstruction Problem

In the Mallow Reconstruction Problem we need to aggregate r noisy orderings $\pi_1, \ldots, \pi_r$ into one optimal ordering π^m. It seems intuitively natural to try to "average" these orderings into one ordering π. It turns out that this intuition is correct, and in fact just taking the average of the locations of element x under the π_i's locates it within a distance of $O(\frac{1}{\beta r} \log n)$ from its location in the true order π^* with high probability. Note that this distance decreases as r is increased.

Somewhat surprisingly, the bulk of the works goes into showing that the optimal ordering π^m is pointwise close to the true ordering π^*. This is important since we want to show that the "average" π is close to π^m, but can only show that it is close to π^*.

Our algorithm uses the "average" order π as a starting point for a dynamic programming algorithm from Section 2 that finds the optimum π^m. The results of this section may be of independent interest in cases where we are looking for an optimum order and have a pointwise good initial guess for it.

1.4.2 Noisy Signal Aggregation

In order to obtain a polynomial time algorithm for the NSA problem it is important to identify that any optimal solution to the problem is close to the true one. Thus the main step of the analysis is the proof of Theorem 11.

To perform the sorting efficiently we use an insertion algorithm. Given an optimal order on a subset of the items we show how to insert a new element. Since the optimal order both before and after the insertion of the element has to satisfy Theorem 11, it is also the case

that no element moves more than $O(\log n)$ after the insertion and re-sorting. Using this we perform a "re-sorting" using the dynamic programming algorithm in Section 2.

The main task is proving Theorem 11 in Section 4.1. We first prove (12) by showing that for a large enough constant c, it is unlikely that any order σ whose total distance from the original order π is more than cn will have $s_q(\sigma) \geq s_q(\pi)$. We then establish (13) in Section 4.1.2 using a bootstrap argument. The argument is based on the idea that if the discrepancy in the position of an element a in an optimal order compared to the original order is more than $c \log n$ for a large constant c, then there must exist many elements that are "close" to a that have also moved by much. This then leads to a contradiction with (12) applied to the neighborhood of a.

The final analysis of the insertion algorithm and the proof of Theorem 10 are provided in Section 4.2. Section 4.3 shows how using a variant of the sorting algorithm it is possible to achieve polynomial running time in sampling complexity $O_\gamma(n \log n)$ thus proving Theorem 12.

It is natural to ask whether the algorithm proposed here is applicable in the more general feedback arc set problem and whether other efficient algorithms for the more general problem are applicable here. It is easy to see that "sorting by number of wins" algorithm, whose approximation ratio has been recently studied [CFR06], will result with high probability with an order σ' with $s_q(\sigma') > n^{3/2-\epsilon} + s_q(\pi)$ for any $\epsilon > 0$ even for a simple Bernoulli q. A similar statement holds for a greedy algorithm where elements are inserted optimally one at a time. With more work it is possible to show that the algorithm presented here does not provide a PTAS for the feedback arc set problem on tournaments and that the complicated algorithm of [KMS07] does not solve the problem presented here.

1.4.3 Comparing the Two Sorting Problems

It is interesting to compare the two sorting problems studied here. The two generative models seem to be very closely related. In fact it is easy to see that if one looks at the random tournament defined by the noisy comparisons model and conditions on it being a permutation, then one recovers the Mallow model. However, the conditioning on the tournament is a very strong conditioning as we condition on an event whose probability is $2^{-\Omega(n^2)}$. This conditioning also has very strong consequences: for example – with constant probability the minimal element in the original π^* will also be the minimal element in the generated order π. Such a property does not hold for the noisy comparisons model as it is easy to see that the probability that the minimal element in π^* will satisfy the maximal number of less equal relations in the noisy input is $n^{-1/2+o(1)}$. In fact, as we will see below, in the noisy order model each generated permutation π satisfies with high probability that $\max |\pi^*(i) - \pi(i)| = O(\log n)$ so in a sense each permutation is already close to the original permutation. For the noisy comparisons problem it is much harder to construct any permutation π satisfying the condition above – and this is one of the main algorithmic challenges we need to overcome.

1.5 Distances between rankings

Here we define a measure of distance between rankings that will be used later, and introduce some notation. First, given two permutations σ and τ we define the *dislocation distance* by

$$d(\sigma, \tau) = \sum_{i=1}^{n} |\sigma(i) - \tau(i)|.$$

Recall that the Kemeny distance $d_K(\sigma, \tau)$ is the number of pairs on which σ and τ disagree. We will write $d(\sigma)$ for $d(\sigma, id)$ where id is the identity permutation and $d_K(\sigma)$ for $d_K(\sigma, id)$. In this paper we will often use the following well known claim [DG77] relating the two distances.

Claim 15. *For any τ,*

$$\frac{1}{2} d(\tau) \leq d_K(\tau) \leq d(\tau).$$

1.6 Acknowledgment

E. M. thanks Andrew Tomkins for inspirational discussions and Marina Meila for interesting discussions on Mallow's model.

2 Sorting an almost sorted list

In this section we present an algorithm that given a pre-sorted list so that each element is at most k positions away from its location in some optimal ordering, finds an optimal ordering in time $O(n \cdot k^2 \cdot 2^{6k})$. The algorithm will be used as a building block for other algorithms in the paper.

Lemma 16. *Let $[n]$ be n elements together with a scoring function q. Suppose that we are given that there is an optimal ordering $\sigma(1), \sigma(2), \ldots, \sigma(n)$, that maximizes the score*

$$s(\sigma) = \sum_{\sigma(i) < \sigma(j)} q(i < j),$$

such that $|\sigma(i) - i| \leq k$ for all i. Then we can find such an optimal σ in time $O(n \cdot k^2 \cdot 2^{6k})$.

In the applications below k will be $O(\log n)$. When k is small ($o(\log n)$), the algorithm tends to linear. Note that a brute force search over all possible σ would require time $k^{\Theta(n)}$. Instead we use dynamic programming to reduce the running time.

Proof. We use a dynamic programming technique to find an optimal sorting. Let $i < j$ be any indices, then by the assumption, the elements in the optimally ordered interval

$$I = \{\sigma(i), \sigma(i+1), \ldots, \sigma(j)\}$$

satisfy $I^- \subset I \subset I^+$ where

$$I^+ = [i-k, j+k], \text{ and } I^- = [i+k, j-k].$$

Hence selecting the set $S_I = \{\sigma(i), \sigma(i+1), \ldots, \sigma(j)\}$ involves choosing a set of size $j-i+1$ that contains the elements of I^- and is contained in I^+. This involves selecting $2k$ elements from the list (or from a subset of the list)

$$[i-k, \ldots, i+k-1] \cup [j-k+1, \ldots, j+k]$$

which has $4k$ elements. Thus the number of such S_I's is bounded by 2^{4k}.

We may assume without loss of generality that n is an exact power of 2. Denote by I_0 the interval containing all the elements. Denote by I_1 the left half of I_0 and by I_2 its right half. Denote by I_3 the left half of I_1 and so on. In total, we will have $n-1$ intervals of lengths $2, 4, 8, \ldots$.

For each $I_t = [i, \ldots, j]$ let S_t denote the possible ($\leq 2^{4k}$) sets of the elements $I'_t = [\sigma(i), \ldots, \sigma(j)]$. We use dynamic programming to store an optimal ordering σ' of each such $I'_t \in S_t$. The total number of I'_t's we will have to consider is bounded by $n \cdot 2^{4k}$. In addition, for each processed interval I'_t we store its optimal score $s'(I'_t, \sigma')$, such that

$$s'(I'_t, \sigma') = \sum_{\sigma'(i')<\sigma'(j'),\ i'<j'<i'+2k} q(i' < j').$$

In other words, we only sum over pairs i', j' in I'_t that are less than $2k$ apart, and which are the only pairs that potentially may get swapped. Note that the actual score $s(I'_t, \sigma')$ is shifted from $s'(I'_t, \sigma')$ by an amount that is independent of σ':

$$s(I'_t, \sigma') = \sum_{\sigma'(i')<\sigma'(j')} q(i' < j') = \sum_{\sigma'(i')<\sigma'(j'),\ i'<j'<i'+2k} q(i' < j') + \sum_{\sigma'(i')<\sigma'(j'),\ j'\geq i'+2k} q(i' < j') = s'(I'_t, \sigma') + \sum_{j'\geq i'+2k} q(i' < j').$$

Hence maximizing $s'(I'_t, \sigma')$ is equivalent to maximizing the actual score $s(I'_t, \sigma')$.

We proceed from $t = n-1$ down to $t = 0$ producing and storing an optimal sort for each possible I'_t. For $t = n-1, n-2, \ldots, n/2$ the length of each I'_t is 2, and the optimal sort can be found in $O(1)$ steps.

Now let $t < n/2$. We are trying to find an optimal sort of a given $I'_t = [i, i+2s-1]$. We do this by dividing the optimal sort into two halves I_l and I_r and trying to sort them separately. We know that I_l must contain all the elements in I'_t that come from the interval $[1, \ldots, i+s-1-k]$ and must be contained in the interval $[1, \ldots, i+s-1+k]$. Thus there are at most 2^{2k} choices for the elements of I_l, and the choice of I_l determines I_r uniquely. For each such choice we look up an optimum solution for I_l and for I_r in the dynamic programming table. Among all possible choices of I_l we pick the best one. This is done by recomputing the score s' for the joined interval, and takes at most $O(k^2)$ time, since the only

new pairs (i', j') with $|i' - j'| < 2k$ are along the boundary between I_l and I_r. Thus the total cost will be

$$\sum_{i=1}^{\log n} \#\text{intervals of length } 2^i \cdot \#\text{checks} \cdot \text{cost of check} =$$

$$\sum_{i=1}^{\log n} O\left(\frac{n \cdot 2^{4k}}{2^i} \cdot 2^{2k} \cdot k^2\right) = O(n \cdot k^2 \cdot 2^{6k}).$$

□

3 Noisy ordering aggregation

We will now turn our attention to aggregating noisy rankings generated by Mallow's model. Recall that in this model, the probability of a permutation π given a true ordering π^* is given by

$$\mathrm{P}[\pi|\pi^*] = \frac{1}{Z(\beta)} e^{-\beta d_K(\pi,\pi^*)}, \tag{14}$$

where $d_K(\pi, \pi^*)$ is the Kemeny distance – the number of pairs which π and π^* order differently. As a first step we show that under this model, locations of individual elements are distributed geometrically.

Lemma 17. *Let a be an element that is ranked k-th by π^*. In other words, $\pi^*(a) = k$. Then*

$$\mathrm{P}[|\pi(a) - k| \geq i] < 2 \cdot e^{-\beta i}/(1 - e^{-\beta}).$$

for all i.

Proof. For simplicity, we assume that π^* is the identity map: $\pi^*(i) = i$. The key observation in the proof is that for any m, the distribution of the locations of $m+1, \ldots, n$ under π remains the same if we condition on the ordering of $\{1, \ldots, m\}$ between themselves under π. Thus π can be sampled by inserting the elements $1, \ldots, n$ into the ordering one-by-one, each time conditioning on the order so far.

Suppose we sampled the relative ordering of $1, \ldots, k-1$ under π, and would like to insert a new element k. By (14), the probability of k being mapped to location $k - i$ is bounded by $e^{-\beta i}$. Note that after further insertions, the location of k may only increase. Hence

$$\mathrm{P}[\pi(k) \leq k - i] < \sum_{j=i}^{\infty} e^{-\beta j} = e^{-\beta i}/(1 - e^{-\beta}). \tag{15}$$

A symmetric argument gives the same bound for $\mathrm{P}[\pi(k) \geq k+i]$, and completes the proof. □

Next, we assume that we are given r independent samples generated by Mallow's model. In each one of them, the location of k is geometrically distributed around k. This allows us to prove a stronger concentration for the average of these locations. Again, for simplicity we assume that π^* is the identity $\pi^*(i) = i$.

Lemma 18. *Suppose that the permutation $\pi_1, \ldots, \pi_r$ are drawn according to (14). Let $a = k$ be the element ranked k-th by π^*. Let $\overline{\pi(a)}$ be the average index of a under the permutations $\pi_1, \ldots, \pi_r$:*

$$\overline{\pi(a)} = \frac{1}{r} \sum_{i=1}^{r} \pi_i(a).$$

Then

$$\mathrm{P}[|\overline{\pi(a)} - k| \geq i] \leq 2 \cdot \left(\frac{(5i+1) \cdot e^{-\beta i}}{1 - e^{-\beta}} \right)^r$$

for all i.

Proof. For a vector $b = (b_1, \ldots, b_r)$ of non-negative integers let A_b denote the event that $\pi_j(a) \leq k - b_j$ for $j = 1, \ldots, r$ for which $b_j > 0$. By (15) we have

$$\mathrm{P}[A_b] < e^{-\beta \sum_{j=1}^{r} b_j} / (1 - e^{-\beta})^r.$$

Next, we note that the event $[\overline{\pi(a)} \leq k - i]$ is covered by

$$\bigcup_{\sum_{j=1}^{r} b_j = r \cdot i} [A_b].$$

Hence

$$P[\overline{\pi(a)} \leq k - i] < \# \left\{ b : \sum_{j=1}^{r} b_j = ri \right\} \cdot \frac{e^{-\beta r i}}{(1 - e^{-\beta})^r} = \binom{ri + r - 1}{r - 1} \cdot \frac{e^{-\beta r i}}{(1 - e^{-\beta})^r} < \frac{(5i+1)^r \cdot e^{-\beta r i}}{(1 - e^{-\beta})^r}.$$

Taking the symmetric bound for $P[\overline{\pi(a)} \geq k + i]$ completes the proof. □

In particular, assuming r is fixed, the following statement holds.

Claim 19. *Let $\alpha > 0$. Then for sufficiently large n,*

$$\mathrm{P}\left[|\overline{\pi(k)} - k| \geq \frac{\alpha + 2}{\beta \cdot r} \log n \text{ for some } k \right] < n^{-\alpha}.$$

Proof. The claim follows immediately from Lemma 18. □

We see that the margin of error for each element decreases proportionally to r. We will now use Lemma 16 from Section 2 to give an efficient algorithm that finds the maximum likelihood permutation π^m given $\pi_1, \ldots, \pi_r$. Recall that such a π^m minimizes

$$\sum_{k=1}^{r} d_K(\pi_k, \pi^m) = \sum_{k=1}^{r} \sum_{\pi^m(i) < \pi^m(j)} 1_{\pi_k(i) > \pi_k(j)} = \sum_{\pi^m(i) < \pi^m(j)} \#\{k : \pi_k(i) > \pi_k(j)\}. \tag{16}$$

Set $q(i<j) := \#\{k : \pi_k(i) < \pi_k(j)\}$. Then minimizing (16) is equivalent to maximizing

$$s(\pi^m) = \sum_{\pi^m(i)<\pi^m(j)} q(i<j).$$

Let $\overline{\pi}$ be the elements $\{k\}$ sorted according to their $\overline{\pi(k)}$ value. By Claim 19 it follows that except with probability $n^{-\alpha}$,

$$|\overline{\pi}(k) - \pi^*(k)| < 2 \cdot \frac{\alpha+2}{\beta \cdot r} \log n \text{ for all } k. \tag{17}$$

In order to apply Lemma 16 to obtain the optimum π^m from the approximation $\overline{\pi}$ it remains to see that with high probability the optimum π^m is pointwise close to the original π^* (and hence, by (17), to $\overline{\pi}$). For simplicity, we assume that π^* is the identity order $1, \ldots, n$.

Denote

$$L = \max\left(6 \cdot \frac{\alpha+2}{\beta \cdot r} \log n,\ 6 \cdot \frac{\alpha+2+1/\beta}{\beta}\right).$$

We first use (15) to prove the following simple claim.

Claim 20. *Except with probability $n^{-\alpha}$ we have that for any i, j such that $i \le j - L$,*

$$q(i<j) > \frac{2}{3}r.$$

In other words, less than $1/3$ of the permutations $\pi_1, \ldots, \pi_r$ order i and j incorrectly.

Proof. By a direct application of (15), for each k,

$$\mathrm{P}[\pi_k(j) < \pi_k(i)] \le \mathrm{P}[\pi_k(j) \le j - L/2] + \mathrm{P}[\pi_k(i) \ge i + L/2] \le 2 \cdot e^{-\beta L/2}/(1-e^{-\beta}) \le n^{-3(\alpha+1)/r},$$

for a sufficiently large n. In the case when $r \le \log n$, the probability of having at least $r/3$ rearranged pairs is bounded by $n^{-(\alpha+1)} \cdot 2^r < n^{-\alpha}$. In the case when $r > \log n$, we have

$$\mathrm{P}[\pi_k(j) < \pi_k(i)] \le e^{-3(\alpha+1)},$$

and the probability of having at least $r/3$ rearranged pairs is bounded by

$$e^{-3(\alpha+1)\cdot r/3} \cdot 2^r < e^{-\alpha \cdot r} < n^{-\alpha}.$$

□

We are now ready to prove the lemma on the proximity of the optimum to the original.

Lemma 21. *Except with probability $< 2 \cdot n^{-\alpha}$, for any optimal π^m and for all k, we have*

$$|\pi^m(k) - \pi^*(k)| \le 32L,$$

where π^* is the original permutation.

Proof. We will assume that the sampled permutations $\pi_1, \ldots, \pi_r$ satisfy the property in Claim 20, which happens except with probability of at most $n^{-\alpha}$. Suppose, for contradiction, that there is a k such that $|\pi^m(k) - k| = M > 32L$. Without loss of generality suppose that $\pi^m(k) = k + M$.

We first claim that there must be at least $T \geq M/4 - L > 7L$ indexes $i < k$ such that $\pi^m(i) \geq k$. That is, many indexes move from below position k to above position k. Let S be the set of indexes j such that $k \leq \pi^m(j) < k + M$. We must have

$$\sum_{j \in S} (q(j < k) - q(j > k)) > 0,$$

for otherwise the permutation π_0^m where k is moved back to location k would score higher than π^m. We spit S into S_1, S_2 and S_3 as follows

$$S = S_1 \cup S_2 \cup S_3 = \{j \in S : \ j < k\} \cup \{j \in S : \ k < j < k + L\} \cup \{j \in S : \ j \geq k + L\}.$$

Note that $|S_2| < L$. Hence, by our assumption,

$$\sum_{j \in S} (q(j < k) - q(j > k)) = \sum_{j \in S_1} (q(j < k) - q(j > k)) + \sum_{j \in S_2} (q(j < k) - q(j > k)) +$$
$$\sum_{j \in S_3} (q(j < k) - q(j > k)) < r \cdot |S_1| + r \cdot |S_2| - (r/3) \cdot |S_3| < r \cdot (T + L) - (r/3) \cdot (M - T - L).$$

Hence $T + L - (M - T - L)/3 > 0$, which implies that $T > 7L$.

The fact that there are T indexes $i < k$ such that $\pi^m(i) \geq k$, implies that there are at least T indexes $i \geq k$ with $\pi^m(i) < k$. Denote

$$T_1 = \{i < k : \pi^m(i) \geq k\}, \ T_2 = \{i \geq k : \pi^m(i) < k\}.$$

Let π_1^m be the permutation obtained from π^m by concatenating its restriction to $H_L = \{1, \ldots, k-1\}$ with its restriction to $H_R = \{k, \ldots, n\}$. We claim that π_1^m, scores higher than π^m, which is a contradiction. We first count the number of pairs $(i < j)$ on which π^m and π_1^m disagree such that $|i - j| < L$. To disagree, either i or j has to belong to $T_1 \cup T_2$, and in each case we have at most L choices for the other. Hence the total number of such pairs is at most $2TL$. We denote these pairs by P_1.

Next we count the number of pairs $(i < j)$ on which π^m and π_1^m disagree such that $|i - j| \geq L$. Note that for each such pair π_1^m has the "right" answer and we know that in this case $q(i < j) > (2/3)r$. Each of the elements of T_1 participates in such a pair with each element of T_2, save at most L elements for which $|i - j| < L$. Thus the number of such pairs is at least $T(T - L)$. We denote them by P_2.

The final difference in score between π^m and π_1^m is given by

$$s(\pi_1^m) - s(\pi^m) = \sum_{(i<j) \in P_1} (q(i < j) - q(j < i)) + \sum_{(i<j) \in P_2} (q(i < j) - q(j < i)) >$$
$$(-r) \cdot |P_1| + (r/3) \cdot |P_2| \geq (-r)(2TL) + (r/3)(T^2 - TL) = r(T^2/3 - 7TL/3) > 0,$$

since $T > 7L$. Contradiction. ∎

It follows from Lemma 21 and Claim 19 that the pointwise distance between $\overline{\pi}$ and π^m is bounded by $k = 33L$. We can now apply Lemma 16 to obtain:

Theorem 7. Let $\pi_1, \ldots, \pi_r$ be rankings on n elements independently generated by Mallow's model with parameter $\beta > 0$, and let $\alpha > 0$. Then a maximum probability order π^m can be computed in time

$$T(n) = O\left(n^{1+O\left(\frac{\alpha}{\beta r}\right)} \cdot 2^{O\left(\frac{\alpha}{\beta}+\frac{1}{\beta^2}\right)} \cdot \log^2 n\right).$$

except with probability $< n^{-\alpha}$. In particular, the algorithm tends to almost linear as r grows.

Remark. It should be noted that since the π_i's are actual orderings, they can be recovered with $O(n \log n)$ queries of the type $j <^?_{\pi_i} k$ each. Thus the total query complexity is trivially bounded by $O(rn \log n)$.

4 Noisy comparisons aggregation

4.1 The Discrepancy between the true order and optimal orders

The goal of this section is to establish that with high probability any optimum solution will not be far from the original solution. We first establish that the orders are close on average, and then that they are pointwise close to each other.

4.1.1 Average proximity

We prove that with high probability, the total difference between the original and any optimal ordering is linear in the length of the interval.

We begin by bounding the probability that a specific permutation σ will beat the original ordering. Recall that $d_K(\sigma)$ is the number of pairs on which the permutation σ disagrees with the identity.

Lemma 22. *Assume that the distributions of the scoring functions are strongly γ-biased, and suppose that the original ordering is $1 < 2 \ldots < n$. Let σ be another permutation. Then the probability that σ beats the identity permutation is bounded from above by*

$$2^{-\gamma d_K(\sigma)}.$$

Proof. In order for σ to beat the identity, it needs to beat it in the $d_K(\sigma)$ positions where they differ. The probability bound follows immediately from the definition of γ-biased distributions. □

Recall that $d(\tau) = \sum_{i=1}^{n} |\tau(i) - i|$ is the total dislocation of elements under τ.

Lemma 23. *The number of permutations τ on $[n]$ satisfying $d(\tau) \leq c\,n$ is at most*

$$2^n\, 2^{(1+c)\,n\,H(1/(1+c))}.$$

Here $H(x)$ is the binary entropy of x defined by

$$H(x) = -x \log_2 x - (1-x)\log_2(1-x) < -2x \log_2 x,$$

for small x.

Proof. Note that each τ can be uniquely specified by the values of $s(i) = \tau(i) - i$, and that we are given that $\sum |s(i)|$ is exactly $d(\tau) \le cn$. Thus there is an injection of τ's with $d(\tau) = m$ into sequences of n numbers which in absolute values add up to m. It thus suffices to bound the number of such sequences. The number of unsigned sequences equals the number of ways of placing m balls in n bins, which is equal to $\binom{n+m-1}{n-1}$. Signs multiply the possibilities by at most 2^n. Hence the total number of τ's with $d(\tau) = m$ is bounded by $2^n \cdot \binom{n+m-1}{n-1}$. Summing up over the possible values of m we obtain

$$\sum_{m=0}^{cn} 2^n \cdot \binom{n+m-1}{n-1} < 2^n \cdot \binom{n+cn}{n} \le 2^n \, 2^{(n+cn)\,H(n/(n+cn))}.$$

□

Lemma 24. *Suppose that the true ordering is $1 < \ldots < n$ and n is large enough. Then if $c \ge 1$ and*

$$\gamma c > 4 \cdot (1 + (1+c)H(1/(1+c))),$$

the probability that any ranking σ is optimal and $d(\sigma) > cn$ is at most $2^{-cn\gamma/5}$ for sufficiently large n. In particular, as $\gamma \to 0$, it suffices to take

$$c = O(\gamma^{-1} \log 1/\gamma) = \tilde{O}(\gamma^{-1}).$$

Proof. Let σ be an ordering with $d(\sigma) > cn$. Then by Claim 15 we have $d_K(\sigma) > cn/2$. Therefore the probability that such an ordering will beat the identity is bounded by $2^{-cn\gamma/2}$ by Lemma 22. We now use union bound and Lemma 23 to obtain the desired result. □

4.1.2 Pointwise proximity

In the previous section we have seen that it is unlikely that the *average* element in the optimal order is more than a constant number of positions away from its original location. Our next goal is to show that the *maximum* dislocation of an element is bounded by $O(\log n)$. As a first step, we show that one "big" dislocation is likely to entail many "big" dislocations.

Lemma 25. *Suppose that the true ordering of $1, \ldots, n$ is given by the identity ranking, that is, $1 < 2 \ldots < n$. Let $1 \le i < j \le n$ be two indices and $m = j - i$. Let A_{ij} be the event that there is an optimum ordering σ such that $\sigma(i) = j$ and the following two conditions hold:*

$$[i, j] \subset \sigma[i - 2m, j + 2m],$$

$$|(\sigma[1, i - \ell - 1] \cup \sigma[j + \ell + 1, n]) \cap [i, j - 1]| \le \ell,$$

i.e., elements from at most $2m$-away are mapped to $[i,j]$ by σ, and at most ℓ elements are mapped to the interval $[i,j-1]$ from outside the interval $[i-\ell,j+\ell]$ by σ. We set $\ell = \lfloor c_\ell m \rfloor < m$, where

$$c_\ell = \frac{\gamma}{300(1-\log\gamma)} = \tilde{\Omega}(\gamma).$$

Then

$$P(A_{ij}) < 2^{-m\gamma/2}.$$

Proof. We prove the lemma by applying a union bound over all possible variants of the set $B = \sigma^{-1}[i,j]$. We know that B may contain a subset of size at most 3ℓ of elements coming from $[i-m,i-1] \cup [j+1,j+m]$, thus the number of possible sets is bounded by

$$\binom{5m}{3\ell} \cdot \binom{m}{3\ell} \leq 2^{5m\cdot H\left(\frac{3\ell}{5m}\right)+m\cdot H\left(\frac{3\ell}{m}\right)} < 2^{6m\cdot H\left(\frac{3\ell}{5m}\right)} < 2^{12m\cdot\frac{3\ell}{5m}\cdot\log\frac{5m}{3\ell}} < 2^{m\gamma/2}.$$

The assumption that σ is optimal implies in particular that moving the i-th element from the j-th position where it is mapped by σ back to the i-th position does not improve the solution. For each specific choice of B, more than $2/3$ of the elements that are mapped to $[i,j-1]$ are originally smaller than i, and hence the probability of moving the i-th element back not improving the solution is bounded by $2^{-m\gamma}$. By union bound,

$$P[A_{ij}] < 2^{m\gamma/2} \cdot 2^{-m\gamma} = 2^{-m\gamma/2}.$$

□

As a corollary to Lemma 25 we obtain the following using a simple union-bound. For the rest of the proof all the log's are base 2.

Corollary 26. *Let*

$$m_1 = (-\log\varepsilon + 2\log n)/(\gamma/2) = O((-\log\varepsilon + \log n)/\gamma),$$

then A_{ij} does not occur for any i,j with $|i-j| \geq m_1$ with probability $> 1-\varepsilon$.

Next, we formulate a corollary to Lemma 24.

Corollary 27. *Suppose that $1 < 2 < \ldots < n$ is the true ordering. Set*

$$m_2 = 2m_1.$$

For each interval $I = [i,\ldots,j]$ with at least m_2 elements consider all the sets S_I which contain the elements from

$$I^- = [i+m_2,\ldots,j-m_2],$$

and are contained in the interval

$$I^+ = [i-m_2,\ldots,j+m_2].$$

Then with probability $> 1 - \varepsilon$ all such sets S_I do not have an optimal ordering that has a total deviation from the true of more than $c_2 |i - j|$, with

$$c_2 = \frac{35}{\gamma} = O(\gamma^{-1}),$$

a constant.

Proof. There are at most $n^2 \cdot 2^{4m_2}$ such sets. The probability of each set not satisfying the conclusion is bounded by Lemma 24 with

$$2^{-c_2 m_2 \gamma/5} = 2^{-7m_2} = 2^{-m_2} \cdot 2^{-2m_2} \cdot 2^{-4m_2} < \varepsilon \cdot n^{-2} \cdot 2^{-4m_2}.$$

The last inequality holds because $m_2 > \max(\log n, -\log \varepsilon)$. By taking a union bound over all the sets we obtain the statement of the corollary. □

We are now ready to prove the main result on the pointwise distance between an optimal ordering and the original.

Lemma 28. *Assuming that the events from Corollaries 26 and 27 hold, it follows that for each optimal ordering σ and for each i, $|i - \sigma(i)| < c_3 \log n$, where*

$$c_3 = \frac{24}{c_\ell^2} \cdot \frac{m_2}{\log n} = \tilde{O}(\gamma^{-3}(-\log \varepsilon / \log n + 1))$$

is a constant. In particular, this conclusion holds with probability $> 1 - 2\varepsilon$.

Proof. We say that a position i is *good* if there is no index j such that $\sigma(j)$ is on the other side of i from j and $|\sigma(j) - j| \geq m_2$. In other words, i is good if there is no "long" jump over i in σ. In the case when $i = j$ or $i = \sigma(j)$ for a long jump, it is not considered good. An index that is not good is bad. An interval I is bad if all of its indices are bad. Our goal is to show that there are no bad intervals of length $\geq c_3 \log n$. This would prove the lemma, since if there is an i with $|i - \sigma(i)| > c_3 \log n$ then there is a bad interval of length at least $c_3 \log n$.

Assume, for contradiction, that $I = [i, \ldots, i + t - 1]$ is a bad interval of length $t \geq c_3 \log n$, such that $i - 1$ and $i + t$ are both good (or lie beyond the endpoints of $[1, \ldots, n]$). Denote by S the set of elements that is mapped to I by σ. Denote the indices in S in their original order by $i_1 < i_2 < \ldots < i_t$, i.e., we have: $\{\sigma(i_1), \ldots, \sigma(i_t)\} = I$.

By the goodness of the endpoints of I we have

$$[i + m_2, i + t - 1 - m_2] \subset \{i_1, \ldots, i_t\} \subset [i - m_2, i + t - 1 + m_2].$$

Denote the permutation induced by σ on S by σ' so $\sigma(i_j) < \sigma(i_{j'})$ is equivalent to $\sigma'(j) < \sigma'(j')$. The permutation σ' is optimal, for otherwise it would have been possible to improve σ by improving σ'.

By Corollary 27 and Claim 15, we have the following bound on the number of switches under σ' (and hence the number of switches on the elements of S between themselves under σ):

$$d_K(\sigma') \leq d(\sigma') \leq c_2 t.$$

In how many switches can the elements of S participate under σ? They participate in switches with other elements of S to a total of $d_K(\sigma')$. In addition, they participate in switches with elements that are not in S. These elements must originate at the margins of the interval I: either in the interval $[i-m_2, i+m_2]$ or the interval $[i+t-1-m_2, i+t-1+m_2]$. Thus, each contributes at most $2m_2$ switches with elements of S. There are at most $2m_2$ such elements. Hence the total number of switches between elements in S and in $\overline{S}$ is at most $4m_2^2$. Hence

$$\sum_{i \in S} |\sigma(i) - i| \leq \sum_{i \in S} \#\{\text{switches } i \text{ participates in}\} \leq 4m_2^2 + 2d_K(\sigma') \leq 4m_2^2 + 2c_2 t. \quad (18)$$

We assumed that the entire interval I is bad, hence for every position i there is an index j_i such that $|\sigma(j_i) - j_i| \geq m_2$ and such that i is in the interval $J_i = [j_i, \sigma(j_i)]$ (or the interval $[\sigma(j_i), j_i]$, depending on the order). Consider all such J_i's. We will say that an interval J_i is *free* if there is no interval J_j intersecting it such that $|J_j| > 2|J_i|$. We will use a Vitali covering lemma argument to show that we can choose a disjoint collection of free intervals whose total length is at least $|I|/5$.

Let $\mathcal{F}$ be the collection of J_i's that are free. We claim that for every $i \in I$ there is an element $J_i \in \mathcal{F}$ such that the "tripling" of J_i: $J_i^3 = [j_i - |J_i|, \sigma(j_i) + |J_i|]$ covers i. We know that there is an interval J_1 that covers i. If J_1 is free, then we are done. Otherwise, there is an interval J_2 that intersects J_1 and is at least twice as long. We continue this process until we reach an interval J_k that is free. How far can i be from the endpoints of J_k? At most

$$|J_{k-1}| + |J_{k-2}| + \ldots + |J_1| < |J_k|.$$

Thus, the tripling of J_k covers i.

The argument now proceeds as follows: Order the intervals in $\mathcal{F}$ in a decreasing length order (break ties arbitrarily). Go through the list and add a J_i to our collection if it is disjoint from all the currently selected intervals. We obtain a collection $J_1, \ldots, J_k$ of disjoint intervals of the form $[j_i, \sigma(j_i)]$. Denote the length of the i-th interval by $t_i = |j_i - \sigma(j_i)| \geq m_2$. Let J_i^5 be the "quintupling" of the interval J_i: $J_i^5 = [j_i - 2t_i, \sigma(j_i) + 2t_i]$. We claim that the J_i^5-s cover the entire interval I. Let m be a position on the interval I. Then there is an interval J in $\mathcal{F}$ such that its tripling J^3 covers m. Choose the longest such interval $J' = [j, \sigma(j)]$. If J' has been selected to our collection then we are done. If not, it means that J' intersects a longer interval J_i that has been selected. This means that the tripling of J' is covered by the quintupled interval J_i^5. In particular, m is covered by J_i^5. We conclude that

$$t = \text{length}(I) \leq \sum_{i=1}^{k} \text{length}(J_i^5) = 5 \sum_{i=1}^{k} t_i.$$

Thus $\sum_{i=1}^{k} t_i \geq t/5$. This concludes the covering argument.

We now apply Corollary 26 to the intervals J_i. Since every J_i is free, we conclude that on an interval J_i the contribution of the elements of S that are mapped to J_i to the sum of deviations under σ is at least ℓ_i^2 where $\ell_i = c_\ell t_i$. Thus

$$\sum_{i \in S} |\sigma(i) - i| \geq \sum_{j=1}^{k} \ell_j^2 = c_\ell^2 \cdot \sum_{j=1}^{k} t_j^2 \geq c_\ell^2 \cdot m_2 \cdot \sum_{j=1}^{k} t_j \geq c_\ell^2 \cdot m_2 \cdot t/5$$
$$\geq m_2 \cdot \frac{c_\ell^2}{6} \cdot c_3 \log n + \frac{c_\ell^2}{30} \cdot m_2 t > m_2 \cdot (4m_2) + 2c_2 t = 4m_2^2 + 2c_2 t,$$

for sufficiently large n. The result contradicts (18) above. Hence there are no bad intervals of length $\geq c_3 \log n$, which completes the proof. □

4.2 The algorithm

We are now ready to give an algorithm for computing the optimal ordering with high probability in polynomial time. Note that Lemma 28 holds for any interval of length $\leq n$ (not just length exactly n). Set $\varepsilon = n^{-\alpha-1}/4$. Given an input, let $S \subset \{1, \ldots, n\}$ be a random set of size k. The probability that there is an optimal ordering σ of S and an index i such that $|i - \sigma(i)| \geq c_3 \log n$, where

$$c_3 = \tilde{O}(\gamma^{-3}(-\log \varepsilon / \log n + 1)) = \tilde{O}(\gamma^{-3}(\alpha + 1)),$$

is bounded by 2ε by Lemma 28. Let

$$S_1 \subset S_2 \subset \ldots \subset S_n$$

be a randomly selected chain of sets such that $|S_k| = k$. Then the probability that an element of an optimal order of any of the S_k's deviates from its original location by more than $c_3 \log n$ is bounded by $2n\varepsilon = n^{-\alpha}/2$. We obtain:

Lemma 29. *Let $S_1 \subset \ldots \subset S_n$ be a chain of randomly chosen subsets with $|S_k| = k$. Denote by σ_k an optimal ordering on S_k. Then with probability $\geq 1 - n^{-\alpha}/2$, for each σ_k and for each i, $|i - \sigma_k(i)| < c_3 \log n$, where $c_3 = \tilde{O}(\gamma^{-3}(\alpha + 1))$ is a constant.*

We are now ready to prove the main result, Theorem 10, which we restate

Theorem 30. *There is an algorithm that runs in time n^{c_4}, where*

$$c_4 = \tilde{O}(\gamma^{-3}(\alpha + 1))$$

is a constant, that outputs an optimal ordering with probability $\geq 1 - n^{-\alpha}$.

Proof. First, we choose a random chain of sets $S_1 \subset \ldots \subset S_n$ such that $|S_k| = k$. Then by Lemma 29, with probability $1 - n^{-\alpha}/2$, for each optimal order σ_k of S_k and for each i, $|i - \sigma_k(i)| < c_3 \log n$. We will find the orders σ_k iteratively until we reach σ_n which will be an optimal order for our problem. Denote $\{a_k\} = S_k - S_{k-1}$. Suppose that we have computed σ_{k-1} and we would like to compute σ_k. We first insert a_k into a location that is close to its original location as follows.

Recall that $c_3 = \tilde{\Theta}(\gamma^{-3}(\alpha+1)) > (\alpha+3)/\gamma$. Break S_k into blocks $B_1, B_2, \ldots, B_s$ of length $c_3 \log n$. We claim that with probability $> n^{-\alpha-1}/2$ we can pinpoint the block a_k belongs to within an error of ± 2, thus locating a_k within $3c_3 \log n$ of its original location.

Suppose that a_k should belong to block B_i. Then by our assumption on σ_{k-1}, a_k is bigger than any element in $B_1, \ldots, B_{i-2}$ and smaller than any element in $B_{i+2}, \ldots, B_s$. By comparing a_k to each element in the block and taking the sum of the comparison scores, we see that the probability of having an incorrect comparison result with a block B_j is bounded by $n^{-\alpha-2}/2$. Hence the probability that a_k will not be placed correctly up to an error of two blocks is bounded by $n^{-\alpha-1}/2$ using union bound.

Hence after inserting a_k we obtain an ordering of S_k in which each element is at most $3c_3 \log n$ positions away from its original location. Hence each element is at most $4c_3 \log n$ positions away from its optimal location in σ_k. Thus, by Lemma 16 we can obtain σ_k in time $O(n^{24c_3+2})$. The process is then repeated.

The probability of each stage failing is bounded by $n^{-\alpha-1}/2$. Hence the probability of the algorithm failing assuming the chain $S_1 \subset \ldots \subset S_n$ satisfies Lemma 29 is bounded by $n^{-\alpha}/2$. Thus the algorithm runs in time $O(n^{24c_3+3}) = n^{\tilde{O}(\gamma^{-3}(\alpha+1))}$ and has a failure probability of at most $n^{-\alpha}/2 + n^{-\alpha}/2 = n^{-\alpha}$. □

4.3 Query Complexity

In this section we outline the proof of Theorem 12. Recall that the theorem states that although the running time of the algorithm is a polynomial of n whose degree depends on γ, the query complexity of a variant of the algorithm is $O(n \log n)$. In this section we demonstrate that our algorithm can be implemented with high probability using only $O(n \log n)$ queries. Note that there are two types of queries in the algorithm. The first type is comparing elements in the dynamic programming, while the second is when inserting new elements. We will show that both parts require only $O(n \log n)$ queries. We start with queries in the dynamic programming part.

Lemma 31. *For all $\alpha > 0, \gamma < 1/2$ there exists $c(\alpha, \gamma) < \infty$ such that the total number of comparisons performed in the dynamic programming stage of the algorithm is at most $c\, n \log n$ except with probability $O(n^{-\alpha}/4)$.*

Proof. Recall that in the dynamic programming stage, each element is compared with elements that are at current distance at most $c_3 \log n$ from it, where $c_3 = c_3(\alpha, \gamma) = \tilde{O}(\gamma^{-3}(\alpha + 1))$.

Consider a random insertion order of the elements $a_1, \ldots, a_n$. Let $S_{n/2}$ denote the set of elements inserted up to the $n/2$-th insertion. Then by standard concentration results it

follows that there exists $c_5(c_3, \alpha)$ such that for all $1 \leq i \leq n - c_5 \log n$ it holds that

$$|[a_i, a_i + c_5 \log n] \cap S_{n/2}| \geq c_3 \log n, \tag{19}$$

and for all $c_5 \log n \leq i \leq n$ it holds that

$$|[a_i - c_5 \log n, a_i] \cap S_{n/2}| \geq c_3 \log n \tag{20}$$

except with probability at most $n^{-\alpha-1}$. Note that when (19) and (20) both hold the number of different queries used in the dynamic programming while inserting the elements from $\{a_1, \ldots, a_n\} \setminus S_{n/2}$ is at most $2c_5 n \log n$, since none of these elements is ever compared to an element that is further than $c_5 \log n$ away from it in the true order.

Repeating the argument above for the insertions performed from $S_{n/4}$ to $S_{n/2}$, from $S_{n/8}$ to $S_{n/4}$ etc. we obtain that the total number of queries used is bounded by:

$$2c_5 \log n(n + n/2 + \ldots + 1) \leq 4c_5 n \log n,$$

except with probability $< n^{-\alpha}/4$. This concludes the proof. □

Next we show that there is implementation of insertion that requires only $O(\log n)$ comparisons per insertion. To this end, we recall condition (b) from Definition 9 of strongly γ-biased distributions.

(b) There is a constant A such that for any A different $\mathcal{D}_{a_k,b_k}$ such that $a_k < b_k$ holds for all the k's,

$$\mathrm{P}\left[\sum_{k=1}^{A} q(a_k < b_k) > 0\right] > 1 - 10^{-3}. \tag{21}$$

Lemma 32. *For all $\alpha > 0$, $A \geq 1$ and $\gamma > 0$ there exists a*

$$C(A, \gamma, \alpha) = \tilde{O}((A\ + \gamma^{-3})(\alpha + 1))$$

such that except with probability $O(n^{-\alpha-2}/2)$ it is possible to perform the insertion in the proof of Theorem 30 so that each element is inserted using at most $C \log n$ comparisons, $O(\log n)$ time and the element is placed a distance of at most $4c_3 \log n$ from its optimal location, as required by the algorithm.

Proof. Bellow we maintain the notation that $c_3(\alpha, \gamma) = \tilde{O}(\gamma^{-3}(\alpha + 1))$ is such that at all stages of the insertion and for each item, the distance between the location of the item in the original order and the optimal order is at most $c_3 \log n$. This will result in an error with probability at most $n^{-\alpha}/2$.

Let $c_6 = O(\alpha + 1)$ be chosen so that

$$\mathrm{P}\left[Bin(c_6 \log n, 0.99) < \frac{c_6}{2} \log n + 2 \log_2 n\right] < n^{-\alpha-3}, \tag{22}$$

Let $c_7 = Ac_6 + 4c_3$.

We now describe an insertion step. Let S denote a currently optimally sorted set. We will partition S into consecutive intervals of length between $c_7 \log n$ and $2c_7 \log n$ denoted $I_1, \ldots, I_t$. We will use the notation I_i' for the sub-interval of $I_i = [s, t]$ defined by $I_i' = [s + 2c_3 \log n, t - 2c_3 \log n]$. We say that a newly inserted element j *belongs* to one of the interval I_i if one of the two closest elements to it in the original order belongs to I_i. Note that j can belong to at most two intervals. An element in S belongs to I_i iff it is one of the elements in I_i. Note furthermore that if j belongs to the interval I_i then its optimal insertion location is determined up to $2(Ac_6 + 6c_3) \log n$. Similarly, if we know it belongs to one of two intervals then its optimal insertion location is determined up to

$$c_8 \log n := 4(Ac_6 + 6c_3) \log n.$$

Note that by the choice of c_3 we may assume that all elements belonging to I_i are smaller than all elements of I_j' if $i < j$ in the true order. Similarly, all elements belonging to I_j are larger than all elements of I_j' if $j > i$. We define formally the interval $I_0 = I_0'$ to be an interval of elements that are smaller than all the items and the interval $I_{t+1} = I_{t+1}'$ to be an interval of elements that is bigger than all items.

We construct a binary search tree on the set $[1, t]$ labeled by sub-intervals of $[1, t]$ such that the root is labeled by $[1, t]$ and if a node is labeled by an interval $[s_1, s_2]$ with $s_2 - s_1 > 1$ then its two children are labeled by $[s_1, s']$ and $[s', s_2]$, where s' is chosen so that the length of the two intervals is the same up to ± 1. Note that the two sub-interval overlap at s'. This branching process terminates at intervals of the form $[s, s + 1]$. Each such node will have a path of descendants of length $c_6 \log n$ all labeled by $[s, s + 1]$.

We use a variant of binary search described in Section 3 of [FPRU90]. The algorithm will run for $c_6 \log n$ steps starting at the root of the tree. At each step the algorithm will proceed from a node of the tree to either one of the two children of the node or to the parent of that node.

Suppose that the algorithm is at the node labeled by $[s_1, s_2]$ and $s_2 - s_1 > 1$. The algorithm will first take A elements from I_{s_1-1}' that have not been explored before and will check that the current item is greater than the majority of them. Similarly, it will make a comparison with A elements from I_{s_2+1}'. If either test fails it would backtrack to the parent of the current node. Note that if the test fails then it is the case that the element does not belong to $[s_1, s_2]$ except with probability < 0.01.

Otherwise, let $[s_1, s']$ and $[s', s_2]$ denote the two children of $[s_1, s_2]$. The algorithm will now perform a majority test against A elements from $I_{s'}$ according to which it would choose one of the two sub-intervals $[s_1, s']$ or $[s', s_2]$. Note again that a correct sub-interval is chosen except with probability at most 0.01 (note that in this case there may be two "correct" intervals).

In the case where $s_2 = s_1 + 1$ we perform only the first test. If it fails we move to the parent of the node. It it succeeds, we move to the single child. Again, note that we will move toward the leaf if the interval is correct with probability at least 0.99. Similarly, we will move away from the leaf if the interval is incorrect with probability at least 0.99.

Overall, the analysis shows that at each step we move toward a leaf including the correct interval with probability at least 0.99. From (22) it follows that with probability at least $1 - n^{-\alpha-3}$ after $c_6 \log n$ steps the label of the current node will be $[s, s+1]$ where the inserted element belongs to either I_s or I_{s+1}. Thus the total number of queries is bounded by $3Ac_6 \log n$.

Now, once we have located the element within $c_8 \log n$ positions, we can refine the search by comparing the element to the relevant blocks B_j from the algorithm in Theorem 30. Thus will take at most $c_8 \log n$ more queries, to a grand total of

$$c_8 \log n + 3Ac_6 \log n = \tilde{O}((A\ + \gamma^{-3})(\alpha + 1))$$

queries to execute the insertion step of the algorithm. This concluded the proof. □

References

[ACN05] N. Ailon, M. Charikar, and A. Newman. Aggregating inconsistent information: ranking and clustering. In *Proceedings of 37th STOC*, 2005.

[Alo06] N. Alon. Ranking tournaments. *Siam Journal on Discrete Mathematics*, 20(1):137–142, 2006.

[BM08] M. Braverman and E. Mossel. Noisy sorting without resampling. In *Proceedings of the nineteenth annual ACM-SIAM symposium on Discrete algorithms (SODA)*, page 268, 2008.

[BTT89] J. Bartholdi, III, C. A. Tovey, and M. A. Trick. Voting schemes for which it can be difficult to tell who won the election. *Soc. Choice Welf.*, 6(2):157–165, 1989.

[CFR06] Don Coppersmith, Lisa Fleischer, and Atri Rudra. Ordering by weighted number of wins gives a good ranking for weighted tournaments. In *SODA '06: Proceedings of the seventeenth annual ACM-SIAM symposium on Discrete algorithm*, pages 776–782, New York, NY, USA, 2006. ACM.

[CSS99] William W. Cohen, Robert E. Schapire, and Yoram Singer. Learning to order things. *J. Artificial Intelligence Res.*, 10:243–270 (electronic), 1999.

[DG77] Persi Diaconis and R. L. Graham. Spearman's footrule as a measure of disarray. *J. Roy. Statist. Soc. Ser. B*, 39(2):262–268, 1977.

[Dia88] Persi Diaconis. *Group representations in probability and statistics*. Institute of Mathematical Statistics Lecture Notes—Monograph Series, 11. Institute of Mathematical Statistics, Hayward, CA, 1988.

[FPRU90] U. Feige, D. Peleg, P. Raghavan, and E. Upfal. Computing with unreliable information. In *Proceedings 22nd STOC*, 1990.

[FV86] M. A. Flinger and J.S. Verducci. Distance based ranking models. *Journal of the Royal Statistical Society B*, 48:359–369, 1986.

[FV88] M. A. Flinger and J.S. Verducci. Multistage ranking models. *J. Amer. Statist. Assoc.*, 83(403):892–901, 1988.

[FV90] M. A. Flinger and J.S. Verducci. Posterior probability for a consensus ordering. *Psychometrika*, 55:53–63, 1990.

[KK07] D. Karp and B. Kleinberg. Noisy binary serach and its applications. In *Proceedings of 11th SODA*, pages 891–890, 2007.

[KMS07] C. Kenyon-Mathieu and W. Schudy. How to rank with few errors. In *Proceedings of 39th STOC*, pages 95–103, 2007.

[Mal57] C. L. Mallows. Non-null ranking models. *Biometrika*, 44:114–130, 1957.

[MPPB07] M. Meila, K. Phandis, A. Patterson, and J. Blimes. Consenus ranking under the exponential model. Preprint, 2007.